An Open-Source Tool for Automated Generation of Black-box xUnit Test Code and its Industrial Evaluation

Christian Wiederseiner, Shahnewaz A. Jolly

Vahid Garousi

Software Quality Engineering Research Group (SoftQual), University of Calgary, Canada

Matt M. Eskandar

MR Control Systems International Inc., Calgary, Canada

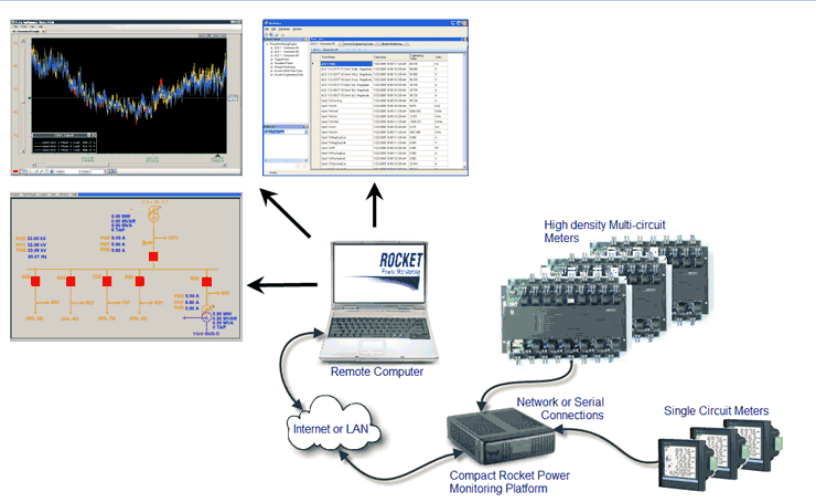
Acknowledging funding and support from:

Talk Outline

- Context - Company and the System Under Test
- Units Under Test (Goal: Unit Testing)
- Black-box Unit Testing (BBUT)
- The Automated BBUT Tool
- Evaluation in an Industrial Setting
- Effectiveness in Detecting Defects
- Conclusions
- Q/A

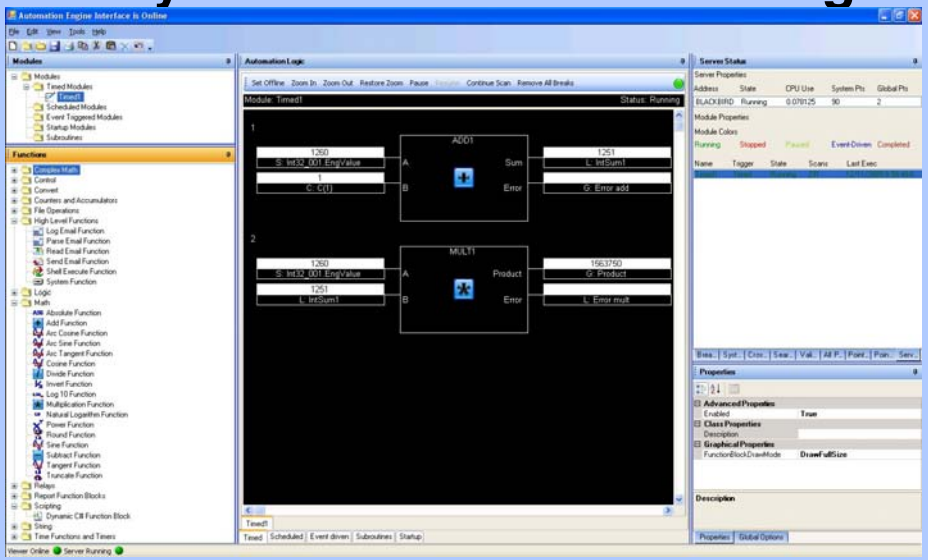
Context - Company and the System Under Test

- A commercial large-scale Supervisory Control and Data Acquisition (SCADA) software system
- Is called *Rocket*
- Has been developed using Microsoft Visual Studio C#
- A total development effort of about 6 man-years.
- Developed using the iterative development process (but not strictly Agile)
- Has now been deployed in several pilot projects and it is well-accepted by the clients.



Context - Company and the System Under Test

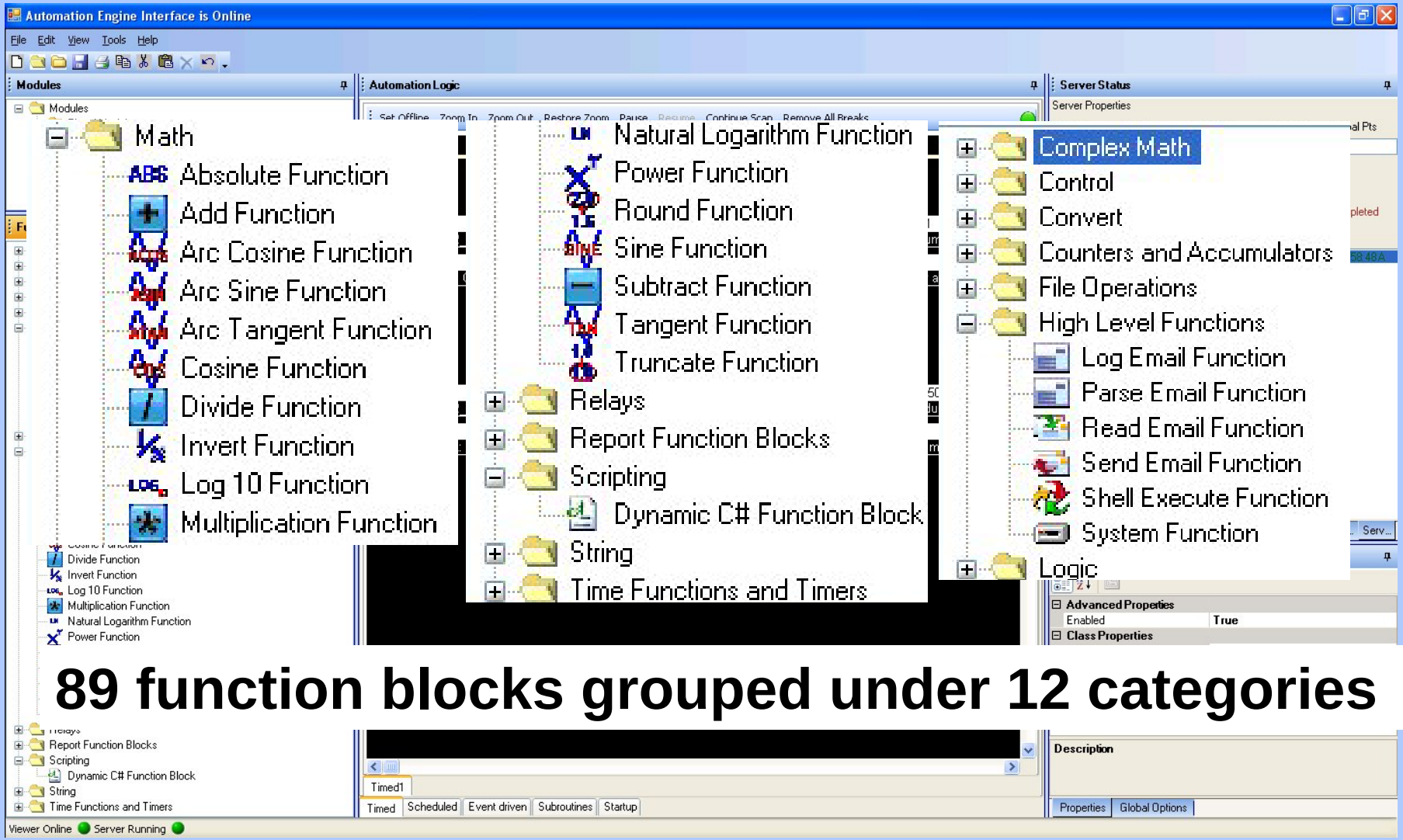
- The SUT has only been tested <u>manually</u> during iterations of the development.
- Towards the end of the project, importance of automated testing was felt
- Thus, a collaboration among the authors
- Our goal: to conduct automated and systematic software testing on the entire Rocket platform.
- Meetings between the project stakeholders
- and prioritizing the modules
- we agreed the first system module to be tested is the *Automation Engine.*



Talk Outline

- Context - Company and the System Under Test
- **Units Under Test (Goal: Unit Testing)**
- Black-box Unit Testing (BBUT)
- The Automated BBUT Tool
- Evaluation in an Industrial Setting
- Effectiveness in Detecting Defects
- Conclusions
- Q/A

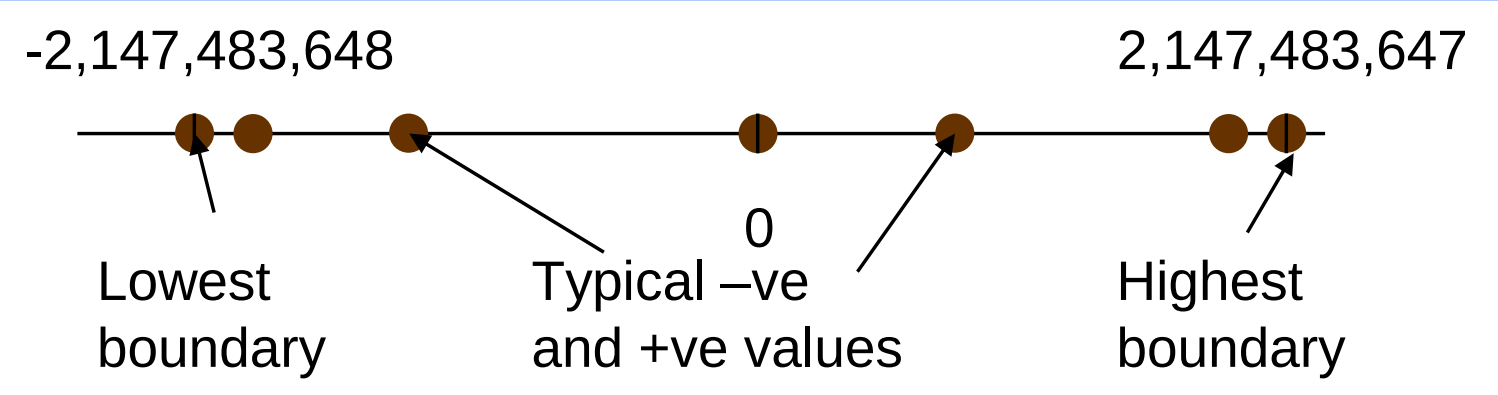
Units Under Test (Goal: Unit Testing)



89 function blocks grouped under 12 categories

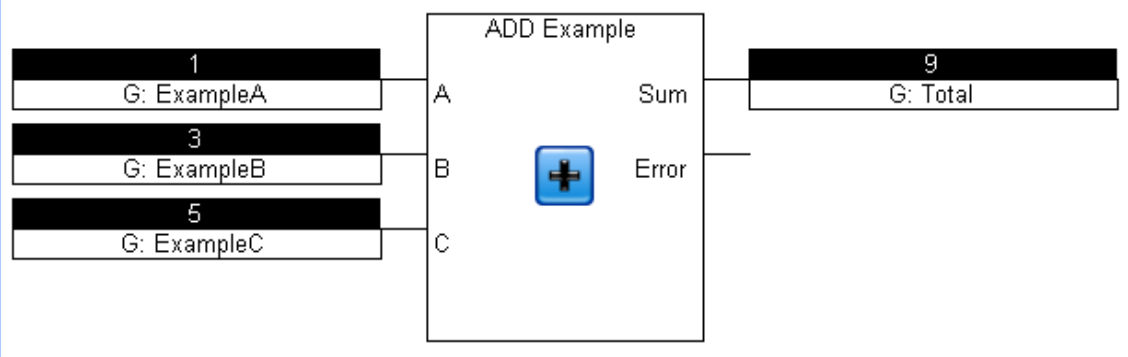
Black-box Unit Testing (BBUT)

Equivalence classes of a 32-bit integer	
1. Nominal positive values	5. Minimum 32-bit integer
2. Nominal negative values	6 .The integer just before the maximum 32-bit integer
3. The value of 0	7. The integer just after the minimum 32-bit integer
4. Maximum 32-bit integer	



Black-box Unit Testing (BBUT): Challenges

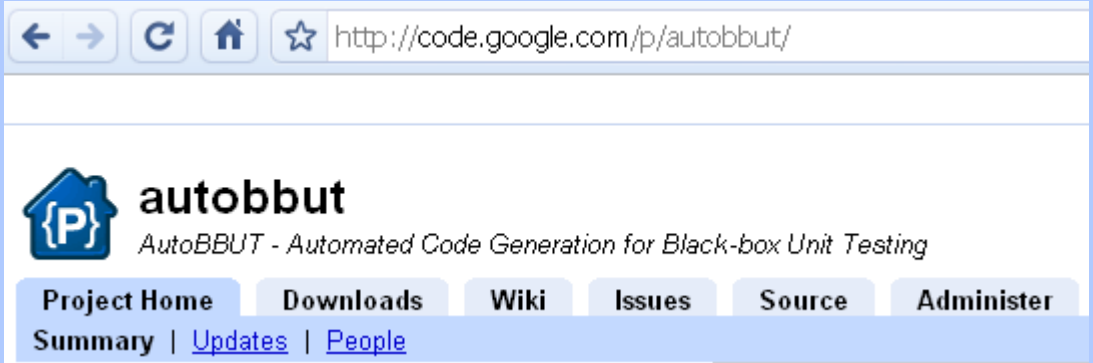
- **The Add function block**



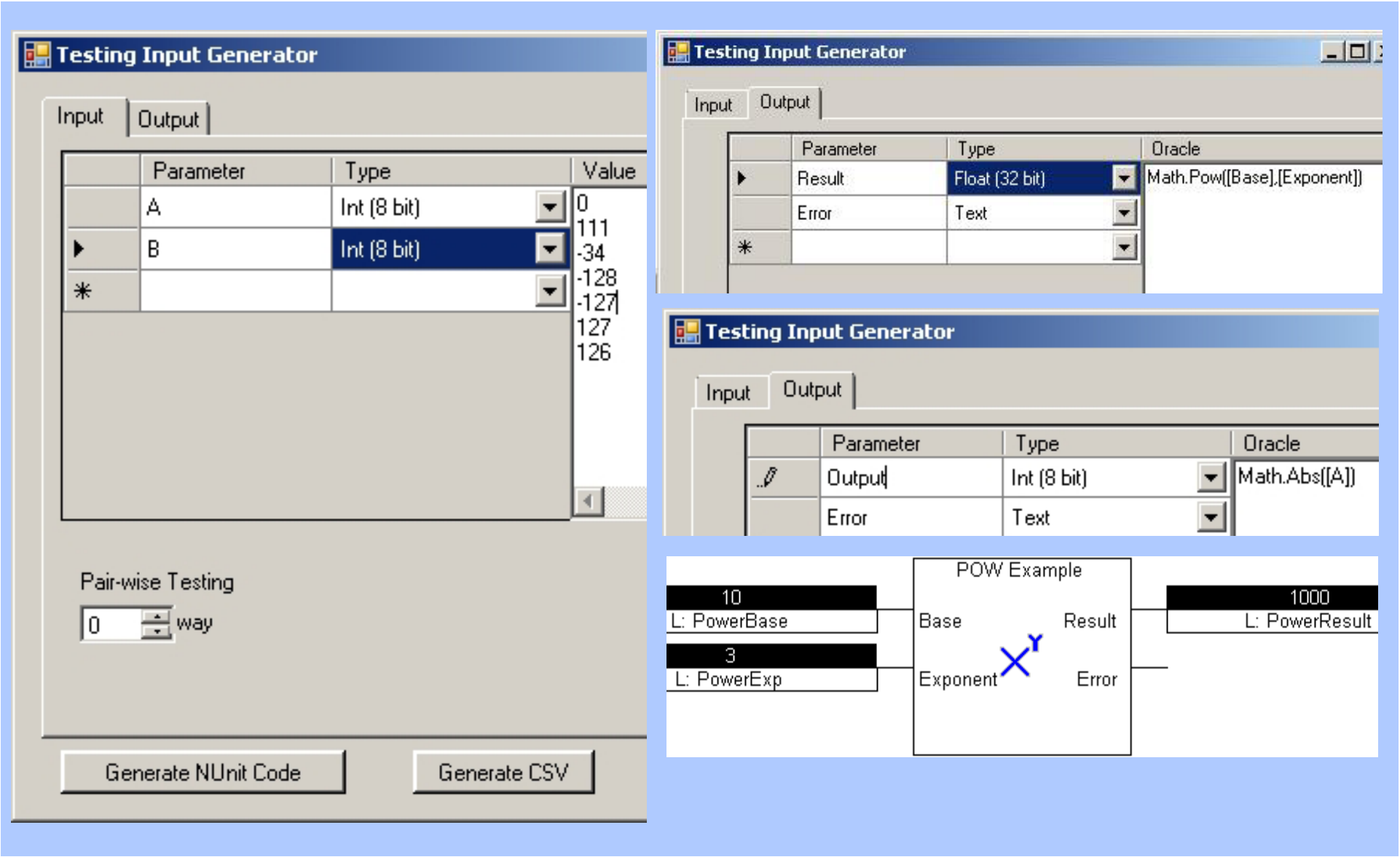
- **If we apply the equivalence classing, we will get 19,683 test cases for only this function block. Bad news ;(**
- **Challenge 1: Coding of test cases (in NUnit): Too much effort**
- **Challenge 2: Coupling of test cases to test input data**
- **Challenge 3: Generation of test oracle**

Black-box Unit Testing (BBUT): Challenges

- **One possible solution → Automated generation of NUnit test code**
- **There are some tools out there:**
  - Microsoft Pex, JML-JUnit, JUB (JUnit test case Builder), TestGen4J, JCrasher, NModel
- **We evaluated those tools for our purpose, but unfortunately, none was suitable (details I the paper)**
- **Decision: to implement our own tool**



AutoBBUT - GUI and Features



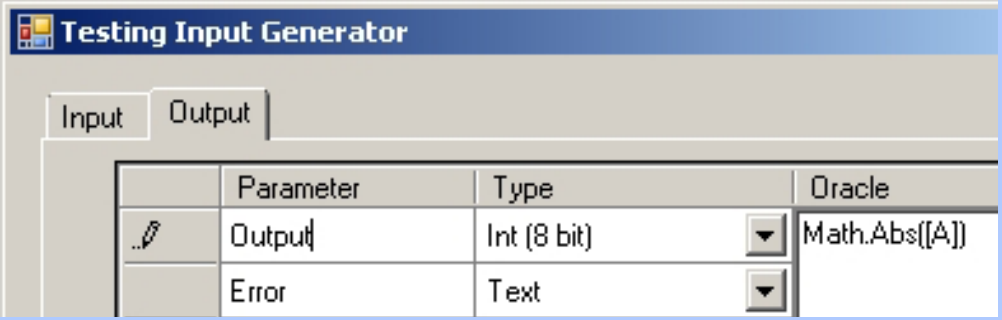
AutoBBUT - Example Usage

TestProject.PowerFBTest | Test61642af496604639a6bc3223c659c542()

```
[TestMethod]
public void Test61642af496604639a6bc3223c659c542()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-128");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-128");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("1.8929E-270"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}

[TestMethod]
public void Test6e134866ed144f3d9c2370bc20cda45f()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-127");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-127");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("-6.5604E-268"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}
```

AutoBBUT – Technologies (Libraries) used

- **To generate all the n-way test cases, we used a recently-introduced Test API from Microsoft, called `Microsoft.Test.VariationGeneration`.**



- **For the development of automated test oracle generation, we used a utility available in the .NET framework class library, called `System.CodeDom.Compiler`.**
- **CodeDOM: Code Document Object Model**

AutoBBUT - Development Details

- **Developed in C# .Net platform. Consists of 875 LOC.**

InputGenerator.TestInputGenerator | generateTestSuiteCode()

```
// loop over all pairwise test cases
foreach (Variation v in combinatoryModel.GenerateVariations(pairwise_way, 1234))
{
    inputList = new Dictionary<string, RocketParameter>();
    Guid guid = Guid.NewGuid();
    // remove -'s from the random string
    string guidstring = Regex.Replace(guid.ToString(), "-", "");
    // start generating the test method one by one
    testSuiteCode += "[TestMethod]" + Environment.NewLine;
    // test method signature
    testSuiteCode += "public void Test" + guidstring + "()" +
        Environment.NewLine + "{" + Environment.NewLine;

    // feeding the input parameters in to the test code
    foreach (Parameter param in combinatoryModel.Parameters)
    {
        RocketParameter input = new RocketParameter();
        testSuiteCode += "TD.setInputParameter(FunctionBlockName,\"" + input.Name + "\", \"" +
            input.Type + "\", \"" + input.Value + "\" );" + Environment.NewLine;
        inputList.Add(input.Name, input);
    }
    testSuiteCode += Environment.NewLine;

    // feeding the output parameters in to the test code
    foreach (DataGridViewRow outputRow in outputGridView.Rows)
```

Talk Outline

- Context - Company and the System Under Test
- Units Under Test (Goal: Unit Testing)
- Black-box Unit Testing (BBUT)
- The Automated BBUT Tool
- **Evaluation in an Industrial Setting**
- Effectiveness in Detecting Defects
- Conclusions
- Q/A

Evaluation in an Industrial Setting

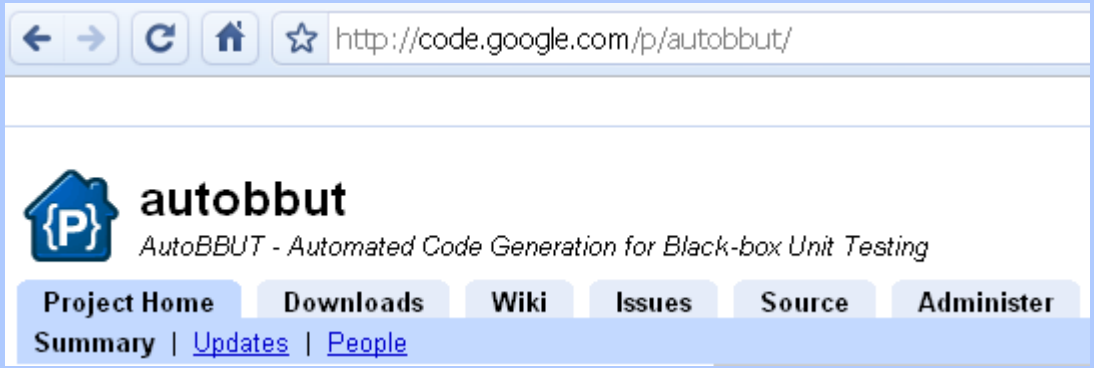
- Only 57 function blocks of 89 function blocks were final in terms of their requirements at this time
- We generated 1,962 NUnit test cases for automated black-box unit testing of 58 function blocks
- The total size of the NUnit test suite is currently 15,906 test LOC.
- Since test case code is automatically generated, each test case method is 8±2 LOC

TestProject.PowerFBTest | Test61642af496604639a6bc3223c659c542()

```
[TestMethod]
public void Test61642af496604639a6bc3223c659c542()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-128");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-128");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("1.8929E-270"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}
```

Conclusions

- **An email from the MRCSI's CEO:**
  - *"Many thanks for your efforts. I reviewed your [defect] report. It looks complete and clear. I and am very pleased with the results. We will include all identified bugs to the list and will try to address them."*



- **Open source**
- **A lot of effort has been spent to have a clean design for it which makes it easily extensible and adaptable to other platforms (e.g., JUnit) and SUTs by other testers.**

Talk Outline

- Context - Company and the System Under Test
- Units Under Test (Goal: Unit Testing)
- Black-box Unit Testing (BBUT)
- The Automated BBUT Tool
- Evaluation in an Industrial Setting
- Effectiveness in Detecting Defects
- Conclusions
- **Q/A**